# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No.: .......................................................................... 09/282,229
Filing Date: ................................................................................. March 31, 1999
Inventors: ........................................................................................... Forin, *et al.*
Appellant: ........................................................................... Microsoft Corporation
Group Art Unit: ............................................................................................... 2194
Examiner: ............................................................................................................ Ho, A.
Confirmation No.: .......................................................................................... 8320
Applicant's Docket No.: ............................................................................. 116650.05
Title: HIGHLY COMPONENTIZED SYSTEM ARCHITECTURE WITH OBJECT MUTATION

## APPEAL BRIEF

To: MS: Appeal Brief - Patent
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

From: David S. Lee
Customer No.: 22971

Sir:

In response to the Final Office Action of August 25, 2005 in connection with the above-identified application, and further to the Notice of Appeal filed on November 23, 2005, an Appeal is made. Favorable consideration is respectfully requested.

## REAL PARTY IN INTEREST

The real party part in interest in the present matter is the Microsoft Corporation of Redmond, WA, USA.

## RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to the Appellant or the Appellant's undersigned representative that would directly affect, or be directly affected by, the outcome of the present Appeal.

## STATUS OF CLAIMS

The Final Office Action of August 25, 2005, states that Claims 1, 2, 4, 6, 7, 9-12, and 14-40 are rejected. This Appeal is made to the rejection of the aforementioned rejected claims.

## STATUS OF AMENDMENTS

Subsequent to the Final Office Action of August 25, 2005, the Appellant filed a Notice of Appeal in the U.S. Patent and Trademark Office on November 23, 2005.

## SUMMARY OF THE INVENTION

The present invention relates to devices and methods for mutating one or more objects during run-time. A summary of the claimed invention may begin with the description set forth on page 61, lines 4 – 20 of the present application:

> An object consists of an interface, an instance pointer, an implementation, and some state. The instance pointers and interfaces are exposed to other objects; the state and the implementation are not…
>
> The preferred embodiment of the invention allows run-time changes to the ordinarily immutable part of an object, even while the object is being used. The term mutation…refers to the act of atomically changing an ordinarily constant part of the object, such as the method implementation. The thread performing the mutation is called a mutator.

Thus, in the example of FIG. 26, for which a description begins on page 63 of the specification, mutation of Object A re-directs interface pointer 2610 from Implementation_A to Implementation_B. More particularly, in FIG. 27, also described beginning on page 63, an object to be mutated includes VTable 2710 that can point to at least one of interfaces 2720 and 2730. Each of interfaces 2720 and 2730 may list different methods that have a pointer 2750 pointing to a different implementation, and mutation within the object may re-direct pointer 2750 to one or more different implementations. Further, mutation of VTable 2710 may re-direct a pointer thereof from interface 2720 to interface 2730, or vice-versa.

## ISSUES

The outstanding issues for appeal include:

1. The rejection of Claims 1, 2, 4, 6, 7, 9 - 12, and 14 - 28 under 35 U.S.C. §103(a) as being unpatentable over Joy (U.S. Patent 5,761,670; hereafter "Joy").

2. The rejection of Claims 29-40 under 35 U.S.C. §103(a) as being unpatentable over Joy in view of Lundin (U.S. Patent 5,339,430; hereafter "Lundin").

## GROUPING OF CLAIMS

The Appellant submits that the claims under appeal do not all stand or fall together. The rejection presented in the Final Office Action is addressed below as it pertains to the following groups of claims.

a. Claim 1;

b. Claims 2, 4, 6, 7, and 9;

c. Claims 10 - 12 and 14;

d. Claims 15-22;

e. Claims 23 - 28;

f. Claims 29 - 31; and

g. Claims 32 - 40.

ARGUMENTS

**a.** It is respectfully submitted that Joy does not teach or suggest mutating an object, as recited in **Claim 1**. More particularly, with regard to Claim 1, Joy does not teach or suggest changing a pointer corresponding to the interface of the object from an identification of a first method in the object to an identification of a second method in the object.

The requirements for establishing a *prima facie* case of obviousness, set forth in MPEP §2143, have not all been met in the final rejection from which this appeal is made. The aforementioned requirements are provided below, and a description the respective deficiencies of the rejection, in accordance with the respective claims, follow.

MPEP §2143 states, in part:

> To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on Appellant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

The computer recited in independent Claim 1 comprises a computer having a memory storing computer-executable instructions supporting plural objects and mutation object, with the mutation object comprising:

- a method for mutating at least one object of said plural objects dynamically during run-time to provide a new implementation within the one object,
- wherein the one object includes a first method and a second method and an interface with a pointer, and
- wherein the method of mutating includes changing the pointer from identification of the first method to identification of the second method.

According to Claim 1, the method for mutating is to mutate at least one of the objects that include a first and second method by changing a pointer within the interface of the object "from identification of the first method to identification of the second method." However, Joy does not teach or suggest changing a pointer corresponding to the interface of the object from an identification of a first method in the object to an identification of a second method in the object. This deficiency of Joy, with respect to Claim 1, is acknowledged on page 3 of the Final Office Action from which this appeal is made as follows, "Joy does not explicitly teach an interface having the pointer."

The Appellant respectfully submits that, further to the fundamental deficiency of failing to teach the interface having a pointer, as is claimed, Joy also fails to provide any motivation for including such a pointer that is changed from identification of a first method to identification of a second method. Instead, Joy describes each method of an object class having its own static pointer, thereby failing to suggest the need for a method of mutating that changes the method to which a pointer points.

More particularly, instead of the claimed object that includes a first method, a second method, and an interface having a pointer, Joy describes an object as follows:

> An object of object class A includes a pointer 204 to the methods for the object...The pointer 204 to the object's methods is actually an indirect pointer to the methods of the associated object class. More particularly, the method pointer 204 points to the Virtual Function Table (VFT) 210 for the object's object class. Each object class has a VFT 210 that includes: (A) pointers 212 to each of the methods 214 of the object class; (B) a pointer 215 to a global lock method (Global Lock1) 216 for synchronizing an object to a thread; (C) a pointer 217 to a special Class object 218; and (D) a pointer 220 to an unlock method 221 for releasing the lock monitors. (Joy, col. 5, lines 12-24; emphasis added by Appellant).

The Appellant respectfully submits that the foregoing description is in conflict with the assertion on page 3 of the Office Action that, "one of ordinary skill in the art would conclude that the virtual function table 210 is in fact the interface contains all of the pointers to each method of the object." That is, *arguendo*, if VFT 210 is akin to the interface recited in Claim 1 because of the presence of method pointers 212 (Joy, col. 5,

line 20) as asserted in the final rejection, Joy still fails to teach or suggest the claimed method that includes, "changing the pointer from identification of the first method to identification of the second method." Instead, the rejection emphasizes that the "method pointer of the object is altered to point to the object-specific VFT" but is silent with regard to "changing the pointer from identification of the first method to identification of the second method," as recited in Claim 1. That is, <u>the pointers to the methods are static according to Joy</u>, shown by the fact that <u>pointers 212 point to each of methods 214 of the object class</u> (Joy, col. 5, line 20). Therefore, there is no need, and thus no motivation, for a method for mutating that includes, "changing the pointer from identification of the first method to identification of the second method," as presently claimed.

Accordingly, Joy is fundamentally deficient with regard to Claim 1 from which the remaining appealed claims depend. Therefore, for at least the reasons set forth above, it is respectfully submitted that the Final rejection fails to establish a *prima facie* case of obviousness.

**b.** Further to the computer of Claim 1, **Claim 2** recites a computer in which each of the plural objects comprises:

- a V-table;
- a V-table pointer pointing to said interface.

The Appellant respectfully submits that Claims 2, 4, 6, 7, and 9 are patentably distinguishable over Joy for at least the reasons set forth in argument (a) above regarding Claim 1.

In addition, the rejection of Claim 1 asserts that one of ordinary skill in the art would conclude that the virtual function table 210 described by Joy is the interface of Claim 1. *Arguendo*, if such assertion is true, then pointers 212 are to point to each of methods 214 of the object class, pointer 215 points to global lock method 216, pointer 217 points to special Class Object 218, and pointer 220 points to an unlock method 221 (Joy, col. 5, lines 20 - 24). However, the final rejection of Claim 2 further

asserts that pointers within virtual function table 210 point to the interface, *i.e.*, virtual function table 210. Thus, the final rejections of Claims 1 and 2 combine to assert that the pointers corresponding to virtual function table 210 point back to virtual function table 210. As shown in FIG. 2 of Joy, clearly none of the pointers point back to virtual function table 210, nor do any of the pointers point to the virtual function table of "One Copy for Object Class 'OBJECT'."

Accordingly, not only do the final rejections of Claims 1 and 2 contradict the actual descriptions and figures provided by Joy, the final rejections also fail to teach or suggest the recitation of Claim 2 that includes a V-table pointer pointing to the interface having a pointer. Therefore, the Appellant respectfully submits that the Final rejection fails to establish a *prima facie* case of obviousness for Claims 2, 4, 6, 7, and 9 over Joy.

**c.** Further to the computer of Claim 1, **Claim 10** recites a computer in which each of the plural objects comprises:

- a state register storing a state of the particular object; and
- wherein the method of the mutation object changes the contents of the state register so as to mutate the state of the particular object.

The Appellant respectfully submits that Claims 10 – 12 and 14 are patentably distinguishable over Joy for at least the reasons set forth in argument (a) above regarding Claim 1.

In addition, Claim 10 recites that <u>each</u> of the plural objects comprises a state register storing a state of the particular object. However, according to Joy, lock data subarray 249 only stores lock data (Joy, col. 6, lines 13 – 15). Thus, for an object that is not locked, "Whether the object is locked or not can be ascertained by checking for the existence of an object-specific locking procedure and an object-specific VFT," (Joy, col. 7, lines 28 – 31). That is, Joy does not teach or suggest the state register for each of the plural objects, as recited in Claim 10.

Therefore, the Appellant respectfully submits that the final rejection fails to establish a *prima facie* case of obviousness for Claims 10 – 12 and 14 over Joy.

**d.** Further to the computer of Claim 1, **Claim 15** recites a computer in which the mutation object further comprises:

- a synchronization of the mutation of one of said plural objects with threads running in said one object.

The Appellant respectfully submits that Claims 15 – 22 are patentably distinguishable over Joy for at least the reasons set forth in argument (a) above regarding Claim 1.

In addition, the final rejection of Claim 15 refers to Joy, col. 4, lines 41 – 54, which defines synchronized methods as using a locking methodology to limit the number of threads that can simultaneously use a system resource. However, Joy provides no further description, and the final rejection provides no further discussion, that even refers to a locking methodology that synchronizes the mutation of an object with threads running in the object, as is claimed.

Thus, Joy fails to teach or suggest the features of Claim 15, and therefore has not established a *prima facie* case of obviousness with regard to Claims 15 – 22 over Joy.

**e.** Further to the computer of Claim 1, **Claim 23** recites a computer, wherein one of the plural objects comprises:

- an interposition object formed by said mutation object mutating a particular one of said plural objects and a copied object at least nearly identical to said one particular object,
- said interposition object differing from said one particular object in that said one particular object has a pointer to said copied object and a

method of interposition between threads seeking said one particular object and said copied object.

The Appellant respectfully submits that Claims 23 – 28 are patentably distinguishable over Joy for at least the reasons set forth in argument (a) above regarding Claim 1.

In addition, the final rejection of Claim 23 refers to the "object of 249 and 260, Fig. 3)" as teaching or suggesting the interposition object of Claim 23. However, the final rejection is vague as what exactly is referred to by "object of 249 and 260." The Appellant submits that neither 240 nor 260 refer to "an interposition object formed by said mutation object mutating a particular one of said plural objects," as claimed. Rather, locked data subarray 249 and locking procedure 260 are part of a data structure 240 for a locked object (Joy, col. 5, lines 58 and 59).

Thus, if the final rejection implies that data structure 240 is to be regarded as the interposition object, the Appellant respectfully submits that data structure 240 simply does not teach or suggest the interposition object as recited in Claim 23. Alternatively, if the final rejection implies that the "object of 249 and 260" is to be regarded as the interposition object, the Appellant further submits that neither the locked data subarray 249 nor the locking procedure 260, as described by Joy, are formed by a mutation object mutating a particular object, as is claimed.

Therefore, in either case, Joy fails to teach or suggest the features of Claim 23, and therefore has not established a *prima facie* case of obviousness with regard to Claims 23 – 28 over Joy.

**f.** Further to the computer of Claim 7, **Claim 29** recites a new implementation corresponding to a software upgrade.

The Appellant respectfully submits that Claims 29 – 31 are patentably distinguishable over Joy for at least the reasons set forth in arguments (a) and (b) above regarding Claims 1 and 2. Further, the Appellant further submits that Lundin

does not compensate for the deficiencies discussed above in arguments (a) and (b), nor does the final rejection make any assertions to that effect. Thus, the proposed combination of Joy and Lundin fails to provide any expectation of success for the linked procedures referred to by Lundin. Accordingly, the final rejection fails to establish a *prima facie* case of obviousness with regard to Claims 29 – 31 over Joy and Lundin.

**g.** Further to the computer of Claim 1, **Claim 32** recites a new implementation corresponding to a different arithmetic algorithm.

The Appellant respectfully submits that Claims 32 – 40 are patentably distinguishable over Joy for at least the reasons set forth in arguments (a) above regarding Claim 1. Further, the Appellant further submits that Lundin does not compensate for the deficiencies discussed above in argument (a), nor does the final rejection make any assertions to that effect. Thus, the proposed combination of Joy and Lundin fails to provide any expectation of success for the linked procedures referred to by Lundin. Accordingly, the final rejection fails to establish a *prima facie* case of obviousness with regard to Claims 32 – 40 over Joy and Lundin.

CONCLUSION

For at least the reasons provided above, it is respectfully submitted that the rejections set forth in the Final Office Action of August 25, 2005, in connection with the subject application should be reversed.

Favorable consideration of this Brief is respectfully requested.

Respectfully submitted,

MICROSOFT CORPORATION

Date: January 6, 2006

David S. Lee
Registration No. 38,222
Direct Telephone: (425) 703-8092

Microsoft Corporation
One Microsoft Way
Attn: Patent Group Docketing Dept.
Redmond, WA 98052

APPENDIX OF CLAIMS ON APPEAL

1. A computer having a memory storing computer-executable instructions supporting plural objects and a mutation object, said mutation object comprising:

a method for mutating at least one object of said plural objects dynamically during run-time to provide a new implementation within the one object, wherein the one object includes a first method and a second method and an interface with a pointer, and

wherein the method of mutating includes changing the pointer from identification of the first method to identification of the second method.

2. The computer of Claim 1 wherein each one of said plural objects comprises:

a V-table;

a V-table pointer pointing to said interface.

4. The computer of Claim 2 wherein said interface comprises a Mutate_Object method.

6. The computer of Claim 2 wherein said mutation object mutates said V-table pointer so as to change the interface of the one object to a new interface corresponding to a new set of methods.

7. The computer of Claim 6 wherein said method of said mutation object is a Mutate_VTable method.

9. The computer of Claim 2 wherein said method of said mutation object is a Mutate_Object method.

10. The computer of Claim 1 wherein each one of said plural objects comprises a state register storing a state of said one object, and wherein said method of said mutation object changes the contents of said state register so as to mutate the state of said one object.

11. The computer of Claim 10 wherein said state register stores the value of a pointer of said one object.

12. The computer of Claim 11 wherein said pointer of said one object comprises a VTable pointer.

14. The computer of Claim 11 wherein said mutation object comprises a Mutate_Object method.

15. The computer of Claim 1 wherein said mutation object further comprises a synchronization of the mutation of one of said plural objects with threads running in said one object.

16. The computer of Claim 15 wherein said synchronization comprises mutual exclusion.

17. The computer of Claim 16 wherein said mutual exclusion prevents new threads from accessing said one object while other threads running in said object are permitted to finish.

18. The computer of Claim 15 wherein said synchronization comprises transactional synchronization.

19. The computer of Claim 18 wherein said transactional synchronization rolls back the threads currently running in the one object and then permits mutation of the object.

20. The computer of Claim 15 wherein said synchronization comprises swizzling.

21. The computer of Claim 20 wherein said swizzling comprises suspending threads running in said one object, mutating the one object and modifying ~~the~~ states of the suspended threads in accordance with the mutation of the one object, and thereafter reactivating the suspended threads.

22. The computer of claim 21 wherein thread states are swizzled between clean points in the thread execution, whereby the threads are suspended at a clean point.

23. The computer of Claim 1 wherein one of said plural objects comprises an interposition object formed by said mutation object mutating a particular one of said plural objects and a copied object at least nearly identical to said one particular object, said interposition object differing from said one particular object in that said one particular object has a pointer to said copied object and a method of interposition between threads seeking said one particular object and said copied object.

24. The computer of Claim 23 wherein said interposition method comprises a filter.

25. The computer of Claim 24 wherein said filter is a read-only filter.

26. The computer of Claim 24 wherein said filter provides access based upon the identity of the requesting thread.

27. The computer of Claim 23 wherein said copied object is a copy of the one particular object.

28. The computer of Claim 27 wherein said interposition object is formed by copying said one particular object and mutating the resulting copy while said copied object is said one particular object.

29. The computer of Claim 7 wherein said new implementation corresponds to a software upgrade.

30. The computer of Claim 7 wherein said new implementation is a higher speed I/O driver.

31. The computer of Claim 7 wherein said new implementation comprises recently loaded code.

32. The computer of Claim 1 wherein said new implementation comprises a different arithmetic algorithm.

33. The computer of claim 1 where said new implementation is a version of an algorithm where specific conditions are assumed to be true, where the version is mutated back to a version when the conditions are no longer true.

34. The computer of claim 33 wherein some of the parameters of the method are assumed to be constant.

35. The computer of claim 34 where the version is generated by a compiler through constant folding.

36. The computer of claim 33 where specific assumptions are made of the objects the method accesses.

37. The computer of claim 36 where the assumption is the location of an object.

38. The computer of claim 36 where the assumption is the value of a field of the state of the object.

39. The computer of claim 36 where the said version is generated through constant folding.

40. The computer of claim 36 where the said version is generated through inlining.